COMPARISON STUDY OF SORTING TECHNIQUES IN DYNAMIC DATA STRUCTURE

ZEYAD ADNAN ABBAS

A dissertation submitted in
partial fulfilment of the requirement for the award of the
Degree of Master of Computer Science (Software Engineering)

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

MARCH 2016

Specially dedicated to…

This project is dedicated to my parents who have supported me all the way since the beginning of my studies and who have been a great source of motivation and inspiration. I also dedicate this work and gave special thanks to the my closest friends

May god bless us always.

ACKNOWLEDGEMENT

First and foremost, I would like to thank the almighty God (ALLAH S.W.T) for all the abilities and opportunities He provided me through all my life and His blessings that enables me to do this research.

It is the greatest honor to be able to work under supervision of Dr. Mohd Zainuri Saringat. I am grateful to be one of the luckiest persons who had a chance to work with him. I am thankful and gratified for all of his help, assistance, inspiration and guidance on the all aspects beside his patience and understanding.

I wish to express my sincere gratitude to everyone who contributed to the successful completion of my study. I would like to express my gratitude to Universiti Tun Hussein Onn Malaysia (UTHM) for all support through my master.

ABSTRACT

Sorting is an important and widely studied issue, where the execution time and the required resources for computation is of extreme importance, especially if it is dealing with real-time data processing. Therefore, it is important to study and to compare in details all the available sorting algorithms. In this project, an intensive investigation was conducted on five algorithms, namely, Bubble Sort, Insertion Sort, Selection Sort, Merge Sort and Quick Sort algorithms. Four groups of data elements were created for the purpose of comparison process among the different sorting algorithms. All the five sorting algorithms are applied to these groups. The worst time complexity for each sorting technique is then computed for each sorting algorithm. The sorting algorithms were classified into two groups of time complexity, $O\ (n^2)$ group and $O(n\log_2 n)$ group. The execution time for the five sorting algorithms of each group of data elements were computed. The fastest algorithm is then determined by the estimated value for each sorting algorithm, which is computed using linear least square regression. The results revealed that the Merge Sort was more efficient to sort data from the Quick Sort for $O(n\log_2 n)$ time complexity group. The Insertion Sort had more efficiency to sort data from Selection Sort and Bubble Sort for $\boldsymbol{O}\ (n^2)$ group. Bubble Sort was the slowest or it was less efficient to sort the data. In conclusion, the efficiency of sorting algorithms can be ranked from highest to lowest as Merge Sort, Quick Sort, Insertion Sort, Selection Sort and Bubble Sort.

ABSTRAK

Isihan merupakan satu masalah yang penting dan telah dikaji secara meluas, di mana masa pelaksanaan dan sumber yang diperlukan untuk membuat pengiraan adalah sangat penting, terutamanya dalam memproses data masa nyata. Oleh itu, ia adalah penting untuk mengkaji dan membandingkan dengan terperinci semua algoritma isihan yang ada. Dalam projek ini, kajian terperinci dijalankan ke atas lima algoritma sisihan, iaitu *Bubble Sort, Insertion Sort, Selection Sort, Merge Sort* dan *Quick Sort*. Empat kumpulan elemen data telah dibentuk untuk membuat perbandingan antara algoritma isihan yang berbeza. Kesemua lima algoritma isihan digunakan kepada kumpulan elemen data ini. Kekompleksan masa bagi setiap algoritma isihan telah dikira. Algoritma isihan dikelaskan kepada dua kumpulan kekompleksan masa iaitu kumpulan $O(n^2)$ dan kumpulan $O(log_2 n)$. Masa pelaksanaan bagi lima algoritma isihan untuk setiap kempulan elemen data telah dikira. Algoritma terpantas kemudian ditentukan menggunakan anggaran nilai untuk setiap algoritma sisihan, yang dikira menggunakan *linear least square regression.* Keputusan menunjukkan bahawa *Merge Sort* lebih pantas untuk menyusun data daripada *Quick Sort* untuk kumpulan kekompleksan masa $O(nlog_2 n)$. *Insertion Sort* lebih cekap untuk menyusun data daripada *Selection Sort* dan *Bubble Sort* untuk kumpulan kekompleksan masa $O(n^2)$. Manakala Bubble Sort mengambil masa paling lama atau paling kurang cekap untuk menyusun data. Kesimpulannya, kecekapan algoritma sisihan boleh disusun mengikut tahap yang paling cekap kepada kurang cekap sebagai *Merge Sort, Quick Sort, Insertion Sort, Selection Sort* dan *Bubble Sor*t.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Sorting is any process of arranging items in any sequence to reduce the cost of accessing the data. Sorting techniques can be divided in two categories; comparison sorting technique and non-comparison sorting techniques. A comparison sort is a type of sorting algorithms that only reads the list of elements through a single abstract comparison operation that determines which two elements should occur first in the final sorted list. Comparison sorting consists of Bubble Sort, Insertion Sort, and Quick Sort. Non-comparison sorting technique does not compare data elements in order to sort them. This category includes Bucket Sort and Count Sort (Verma, & Kumar, 2013). The sorting algorithm is used to rearrange the items of a given list in any order. Though dozens of sorting algorithm have been developed, no single sorting technique is best suited for all applications. Great research went into this category of algorithm because of its importance. These types of algorithm are very much used in many computer algorithms. For instance, searching an element, database algorithm and many more. Sorting algorithm can be classified into two categories, internal sorting and external sorting. In internal sorting all items to be sorted are kept in the main memory. External sorting, on the other hand, deals with a large number of items, hence have resided in auxiliary storage devices such as tape or disk (Dutta, 2013).

Weiss (2003) stressed on the importance of sorting. For instance, the words in a dictionary must be sorted and case distinctions must be ignored. The files in a directory must be listed in sorting order. The index of a book must be sorted and case distinctions must ignored. Card catalog in a library must be sorted by both author and title. A listing

of course offerings at a university must be sorted, first by department and then by course number. Many banks provide statements that list cheques in increasing order by cheque number. In a newspaper, the calendar of events in a schedule is generally sorted by date. The musical compact disks in a record store are generally sorted by recording artist. In the programs printed for graduation ceremonies, departments are listed in sorted order and then students in those departments are listed in sorted order.

Many computer scientists consider sorting to be a fundamental problem in the study of algorithms. There are several reasons (Cormen *et al.*, 2009). First, sometimes they need to sort information that is inherent to an application. For example, in order to prepare customer statements, the bank needs to sort cheques by cheque number. Second, the algorithms often use sorting as a key subroutine. For example, a program that renders graphical objects that are layered on top of each other might have to shoot the object according to the layered relation so that it can draw these objects from bottom to top. Third, sorting is a problem for which can achieve a nontrivial lower bound. The best upper bound must match with lower bound asymptotically so know that the sorting algorithm is optimal. Fourth, there are a wide variety of sorting algorithms and they use various techniques. In fact, there are many important techniques used throughout the algorithm design that have been developed over the years. Therefore, sorting is a problem of historical interest. Previously, many engineering issues come to the fore when they are implementing sorting algorithms. The fastest sorting program for a particular situation may depend upon many factors, such as prior knowledge about the keys and data, the memory hierarchy of the host computer and the software environment.

The linked list data structure is one of the dynamic data structure methods that are used to store the data. The linked list is the data structure used to save unsorted elements and it was used from the sorting techniques namely, Bubble Sort, Insertion Sort, Selection Sort, Merge Sort and Quick Sort to create a sorted linked list.

Thus, this study was comparing the sorting techniques according to worst case time complexity and sorting execution time results.

1.2 Problem Statement

Sorting is the most fundamental algorithmic problem in computer science and a rich source of programming problems for two distinct reasons. First, sorting is a useful operation which efficiently solves many tasks that every programmer encounters. Second, literally dozens of different sorting algorithms have been developed, each of which rests on a particular clever idea or observation (Swain *et al.*, 2013). Therefore, it is essential to compare the prominent of sorting algorithm that is used in recent technology. Today, rapid technology has enhanced the requirement for data processing to meet the demand of the next era. Internet of things (IOT) (Vermesan, & Friess, 2013) is one of the factors that increased the row of data significantly. Therefore, it is important to use a sorting algorithm efficiently. However, it is unknown which algorithm provides the optimum result, considering the required execution time for each algorithm. This research was compare sorting algorithms in order to decide the proper algorithm to avoid the hardware requirement which saves time and reduced the cost significantly. Bubble Sort, Insertion Sort, Selection Sort, Quick Sort and Merge Sort are a sample of sorting techniques that were compared in the research to determine which sorting techniques that are sufficient to sort different sizes of data. In this research, the difference, similarity and working for these sorting techniques was explained in details.

1.3 Research Objectives

The objectives of this project are to:

(i) implement Selection Sort, Insertion Sort, Bubble Sort, Quick Sort and Merge sort using dynamic data structure (Linked list).

(ii) apply four groups of data elements into those techniques in (i).

(iii) calculate the time complexity, such as the execution time and worst case time complexity.

1.4 Research Scope

This study focuses on the problem of sorting of many data sizes to determine which sorting technique provides a result faster than other techniques in terms of the execution time. They are grouped into:

(i) first group from 100 to 1000 data elements
(ii) second group from 2000 to 10000 data elements
(iii) third group from 11000 to 20000 data elements
(iv) fourth group from 21000 to 30000 data elements

A linked list dynamic data structure is used to store the data. Five Sorting algorithms are used, namely, Selection Sort, Insertion Sort, Bubble Sort, Merge Sort and Quick Sort. The execution time is the criteria used to discuss the performance of these sorting algorithms. The Visual C++ program is used to implement the sorting algorithm program and calculate the execution time for all sorting algorithms.

1.5 Outline of Report

This research consists of five chapters. Chapter 1 covers the overview and main objectives of the research. It also consists of the scope of work covered. Chapter 2 illustrates the Selection Sort and Insertion Sort, Merge Sort, Quick Sort and Bubble Sort. This chapter focus on literature review and related works of the research. Chapter 3 discusses the methodology used to achieve the entire objectives of this research. Chapter 4 explains the implementation and the detailed steps as well as the results and discussion. Finally, Chapter 5 illustrates objectives achieved, future work and conclusion

.

1.6 Chapter Summary

This chapter contains explanations about all the techniques required in this research and also addressed the problem and the importance of sorting techniques as well as how to find the fastest sorting technique. This research includes the application of techniques of sorting data stored in linked lists and calculate the execution time of each sort techniques, and then performs the comparison among these techniques. The techniques that will be applied in this research are the Insertion Sort, Selection Sort, Bubble Sort, Merge Sort and Quick Sort, which differ from one another in terms of how they function.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter will explain the literature review of sorting techniques such as Selection Sort, Insertion Sort and Bubble Sort. These sorting techniques were implemented in dynamic data structure (Linked list). All sorting techniques were used during the research steps and related works will be explained. Based on the research objectives, there are various methods that were used to find the best sort technique to sort data. This chapter explains these methods in the comparison study between the sorting techniques.

2.2 Data Structure

A data structure is the implementation of an abstract data type (ADT). The term data structure often refers to data stored in a computer's main memory. An ADT is the realization of a data type as a software component. The interface of the ADT is defined in terms of a type and a set of operations on that type. The behavior of each operation is determined by its inputs and outputs. An ADT do not specify how the data type is implemented. A data type is a type together with a collection of operations to manipulate the type. For example, an integer variable is a member of the integer data type and an addition is an example of an operation of the integer data type. A data item is a piece of information or a record whose value is drawn from a type. A data item is said to be a member of a type. A type is a collection of values. For example, the Boolean type consists of the values true and false. An integer is a simple type because its values

contain no subparts. A record is an example of an aggregate type or composite type. Encapsulation is an implementation details that is hidden from the user of the ADT and protected from outside access (Shaffer, 2010). A linked list is a data structure consisting of a group of nodes which together represent a sequence. Under the simplest form, each node is composed of a datum and a reference (in other words, a link) to the next node in the sequence; more complex variants add additional links. This structure allows for efficient insertion or removal of elements from any position in the sequence (De L'Armee & Contax, 2012).

Dynamic data structures can increase and decrease in size while the program is running. The advantages of dynamic data structures is that it only use the space that is needed at any time. It makes efficient use of the memory and the storage no longer required can be returned to the system for other uses. Meanwhile, the disadvantages of Dynamic Data Structures are difficult to program, can be slow to implement searches, and only allows serial access for a linked list (Alicia Sykes, 2012).

A linked list is a collection of components, called nodes. Every node except the last node contains the address of the next node. Thus, every node in a linked list has two components, one to store the relevant information that is a data and one to store the address, called the link, of the next node in the list. The address of the first node in the list is stored in a separate location, called the head or first. Figure 2.1 is a pictorial representation of a node.

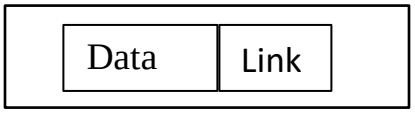

Figure 2.1: Structure of node

Linked lists are a list of items, called nodes, in which the order of the nodes is determined by the address, called the link, stored in each node. The list in Figure 2.2 is an example of a linked list.

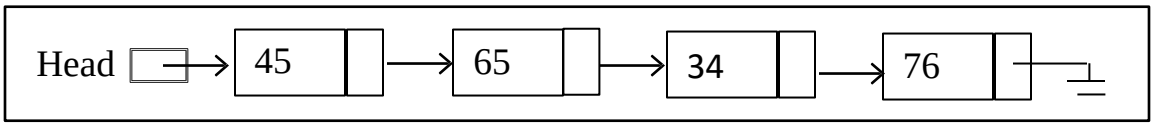

Figure 2.2: Linked list example

The arrow in each node indicates that the address of the node to which it is pointing is stored in that node. The down arrow in the last node indicates that this link field is NULL (Malik, 2010). The advantages and disadvantages of linked list will be explained in the next two subsections.

2.2.1 Advantages of Linked list

The linked list advantages can be summarized in the following steps:

i. It is not necessary to know in advance the number of elements to be stored in the list and therefore, need not allocated as and when necessary.
ii. In a linked list, insertions and deletions can be handled efficiently without fixing the size of the memory in advance.
iii. An important advantage of linked lists over arrays is that the linked list uses exactly; as much memory as it needs, and can be made to expand to fill all available memory locations if needed (Chauhan, 2012).

2.2.2 Disadvantages of Linked list

The disadvantages of linked list can be summarized in the following two steps:

i. The traversal is sequential.
ii. Increased overhead for storing pointers for linking the data items (Vivik, 2012).

2.3 Sorting Techniques

Sorting means rearranging data in any order, such as ascending; or descending with numerical data or alphabetically with character data (Dutta, 2013). Sorting is a commonly used operation in computer science. In addition to its main job, sorting often require to facilitation of some other operation such as searching, merging and normalization (Renu & Manisha, 2015). There are many sorting algorithms that are being used in practical life as well as in computation. A sorting algorithm consists of comparison, swap, and assignment operations (Khairullah, 2013). The usefulness and significance of sorting is depicted from day to day application of (Kapur *et al.*, 2012). Sorting of real life objects, for instance, the telephone directories, incoming tax files, tables of contents, libraries and dictionaries.

In the following subsection, five sorting algorithms, such as, Bubble Sort, Insertion Sort, Selection Sort, Merge Sort and Quick Sort will be explained in some of the details.

2.3.1 Selection Sort

In Selection Sort, a list is sorted by selecting elements in the list, one at a time, and moving them to their proper positions. This algorithm finds the location of the smallest element in the unsorted portion of the list and moves it to the top of the unsorted portion (that is, the whole list) of the list. The first time locates the smallest item in the entire list; the second locates the smallest item in the list starting from the second element in the list, and so on. Selection Sort is a sorting algorithm, specifically an in-place comparison sort. It has an $O(n^2)$ complexity, making it inefficient on large lists, and generally performs worse than the similar Insertion Sort. Selection Sort is noted for its simplicity and also has performance advantages over more complicated algorithms in certain situations (Mishra, 2009). For example supposes the list shown in Figure 2.3.

REFERENCES

Alicia Sykes (2012). *Data Structure and Manipulation.* Retrieved on April 6, 2015, from http://a2computing.as93.net/downloads/Data%20Structures%20and%20Data%20Manipulation%20(Everything).pdf.

Aliyu, A. M. and Zirra, P. B. (2013). A Comparative Analysis of Sorting Algorithms on Integer and Character Arrays. *The International Journal of Engineering and Science*. 2(7): 25-30.

Brown, C. (1972). Merge Sort algorithm [M1]. *Communications of the ACM*. *15*(5), 357-358.

Beniwal, S. and Grover, D. (2013). Comparison of various sorting algorithms: A Review. *International Journal of Emerging Research in Management and Technology*. 2(5): 83-86.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. *Introduction to Algorithms*. 3rd Ed. Cambridge: MIT Press. 2009.

De L'Armee, M. and Contax, T. *Fundamental Data Structures*. International Edition. Cambridge: Rick Miller. 2012.

Dutta, P. S. (2013). An Approach to Improve the Performance of Insertion Sort Algorithm. *International Journal of Computer Science & Engineering Technology*. 4(5): 503-505.

Drozdek, A. *Data Structures and algorithms in C++*. 4th Ed. Thomson Learning: Brooks/ Cole. 2012.

EL Abbadi, N. K. and Kaeem, Z. Y. A. (2014). NK-Sorting Algorithm. *Journal of Kufa for Mathematics and Computer*. 1(4): 27-35.

Horsmalahti, P. (2012). Comparison of Bucket Sort and RADIX Sort. *arXiv preprint arXiv*.1(15): 1-10.

Hoare, C. A. (1962). Quicksort. *The Computer Journal*. *5*(1), 10-16

Jadoon, S., Solehria, S. F.and Qayum, M. (2011). Optimized Selection Sort Algorithm is Faster than the Insertion Sort Algorithm: a Comparative Study. *International Journal of Electrical & Computer Sciences*.11(02): 19-24,

Jariya Phongsai (2009). *Research paper on Sorting Algorithm*. Retrieved on May 6, 2015, from http:// www. eportfolio. lagcc. cuny. edu/ scholars/ doc_fa09/ eP_fa09/Jariya.%20Phongsai/documents/mac%20286/sorting%20algorithms%20 research.pdf.

Jain, D. C. (2014). A Comparative Study on Different Types of Sorting Algorithms (On the Basis of C and Java). *International Journal of Computer Science & Engineering Technology*. 5(08): 805-808.

Khairullah, M. (2013). Enhancing Worst Sorting Algorithms. *International Journal of Advanced Science and Technology*. 56(1): 13-26.

Karunanithi, A. K. A. Survey Discussion and Comparison of Sorting Algorithms. Master's Thesis. *Umea˚ University*; 2014.

Kapur, Eshan, Kumar, Parveen, Gupta and Sahil (2012). Proposal of a two way sorting Algorithm and performance Comparison with existing Algorithms. Interna*tional Journal of Computer Science, Engineering and Applica*. 2(3): 61-78.

Malik, D. S. *Data structures using C++*. 2nd Ed. Course Technology : Cengage Learning. 2010.

Mishra, A. D. *Selection of Best Sorting Algorithm for a Particular problem*. Ph.D. Thesis. Thapar University Patiala; 2009.

Mark A. Weiss. *Data Structures Algorithms Analysis in C++*. 3rd Ed. Pearson Education: Addison-Wesley. 2006.

Moore, D. S. *The basic practice of statistics*. 5rd Ed. New Work: W.H. Freeman and Company. 2010.

Miss. Pooja K. Chhatwani, Miss. Jayashree S. and Somani, (2013). Comparative Analysis Performance of Different Sorting Algorithm in Data Structure. *Internationa Journal of Advanced Research in Computer Science and Software Engineering.* 3(11): 500-507.

Ocampo, J. P. (2008). *An empirical comparison of the runtime of five sorting* Algorithms. Retrrived on December 16, 2015, from http://uploads.julianapena.com/ib-ee.pdf.

Rao, D. D. and Ramesh, B. (2012). Experimental Based Selection of Best Sorting Algorithm. *International Journal of Modern Engineering Research*. *2*(4), 2908-2912.

Renu and Manisha. (2015). MQ Sort an Innovative Algorithm using Quick Sort and Merge Sort. *International Journal of Computer Applications*. 122(21):10-14

Shene, C. K. (1996). A comparative study of linked list sorting algorithms. *3C ON-LINE*, *3*(2): 4-9.

Sareen, P. (2013). Comparison of sorting algorithms (on the basis of average Case). *International Journal of Advanced Research in Computer Science and Software Engineering*. 3(3): 522-532.

Swain, D., Ramkrishna, G., Mahapatra, H., Patr, P. and Dhandrao, P. M. A. (2013). Novel Sorting Technique to Sort Elements in Ascending Order. *International Journal of Engineering and Advanced Technology*. 3(1): 212-126.

Sipser and Michael. *Introduction to the Theory of Computation*. 1st Ed. Thomson: Course Technology. 2006.

Shaffer, C. A. *A Practical Introduction to Data Structures and Algorithm Analysis*. 3rd Ed. Blacksburg: Prentice Hall. 2010.

Vermesan, O., and Friess, P. (Eds.). *Internet of things: converging technologies for smart environments and integrated ecosystems*. 1st Ed. Aalborg: River Publishers. 2013.

Vivek Chauhan, 2012, *linked list advantages and disadvantages*. Retrieved on November 4, 2015 from http://solvewithc.blogspot.my/2012/04/linked-list-advantages-and.html

Verma, A. K., & Kumar, P. (2013). List Sort: A New Approach for Sorting List to Reduce Execution Time**.** *arXiv preprint arXiv*. 1(26): 10-17.

Weiss, M. A. *Data Structure and problem solving using* C++.2nct Ed. Addison Wesly: Pearson. 2003.